

Python: module cdms.cdxmllib

[cdms.cdxmllib](#)

[index](#)

A parser for XML, using the derived class as static DTD.

Modules

[re](#)

[string](#)

Classes

[XMLParser](#)

[TestXMLParser](#)

[exceptions.RuntimeError\(exceptions.StandardError\)](#)

[Error](#)

class [Error\(exceptions.RuntimeError\)](#)

Method resolution order:

[Error](#)

[exceptions.RuntimeError](#)

[exceptions.StandardError](#)

[exceptions.Exception](#)

Methods inherited from [exceptions.Exception](#):

[__getitem__\(...\)](#)

[__init__\(...\)](#)

[__str__\(...\)](#)

class [TestXMLParser\(XMLParser\)](#)

Methods defined here:

[__init__\(self, **kw\)](#)

[close\(self\)](#)

[flush\(self\)](#)

```
handle_cdata(self, data)

handle_comment(self, data)

handle_data(self, data)

handle_doctype(self, tag, pubid, syslit, data)

handle_proc(self, name, data)

handle_xml(self, encoding, standalone)

syntax_error(self, message)

unknown_charref(self, ref)

unknown_endtag(self, tag)

unknown_entityref(self, ref)

unknown_starttag(self, tag, attrs)
```

Methods inherited from [XMLParser](#):

```
feed(self, data)
    # Interface -- feed some data to the parser. Call this as
    # often as you want, with as little or as much text as you
    # want (may include '\n'). (This just saves the text, all th
    # processing is done by goahead().)
```

```
finish_endtag(self, tag)
    # Internal -- finish processing of end tag
```

```
finish_starttag(self, tagname, attrdict, method)
    # Internal -- finish processing of start tag
```

```
getnamespace(self)
    # Interface - return a dictionary of all namespaces currently
```

```
goahead(self, end)
    # Internal -- handle data as far as reasonable. May leave st
    # and data to be processed by a subsequent call. If 'end' is
    # true, force handling all data as if followed by EOF marker.
```

```
handle_charref(self, name)
    # Example -- handle character reference, no need to override
```

```
handle_endtag(self, tag, method)
    # Overridable -- handle end tag
```

```
handle_starttag(self, tag, method, attrs)
    # Overridable -- handle start tag
```

```

parse_attributes(self, tag, i, j)
    # Internal -- parse attributes between i and j

parse_cdata(self, i)
    # Internal -- handle CDATA tag, return length or -1 if not terminated

parse_comment(self, i)
    # Internal -- parse comment, return length or -1 if not terminated

parse_doctype(self, res)
    # Internal -- handle DOCTYPE tag, return length or -1 if not terminated

parse_endtag(self, i)
    # Internal -- parse endtag

parse_proc(self, i)
    # Internal -- handle a processing instruction tag

parse_starttag(self, i)
    # Internal -- handle starttag, return length or -1 if not terminated

reset(self)
    # Interface -- reset this instance. Loses all unprocessed data

setliteral(self, *args)
    # For derived classes only -- enter literal mode (CDATA)

setnomoretags(self)
    # For derived classes only -- enter literal mode (CDATA) till end

translate_references(self, data, all=1)
    # Interface -- translate references

```

Data and other attributes inherited from [XMLParser](#):

attributes = { }

elements = { }

entitydefs = {'amp': '&', 'apos': ''', 'gt': '>', 'lt': '<', 'quot': '"'}

class [XMLParser](#)

Methods defined here:

```

__init__(self, **kw)
    # Interface -- initialize and reset this instance

close(self)
    # Interface -- handle the remaining data

```

```

feed(self, data)
    # Interface -- feed some data to the parser. Call this as
    # often as you want, with as little or as much text as you
    # want (may include '\n'). (This just saves the text, all th
    # processing is done by goahead().)

finish_endtag(self, tag)
    # Internal -- finish processing of end tag

finish_starttag(self, tagname, attrdict, method)
    # Internal -- finish processing of start tag

getnamespace(self)
    # Interface -- return a dictionary of all namespaces currently

goahead(self, end)
    # Internal -- handle data as far as reasonable. May leave st
    # and data to be processed by a subsequent call. If 'end' is
    # true, force handling all data as if followed by EOF marker.

handle_cdata(self, data)
    # Example -- handle cdata, could be overridden

handle_charref(self, name)
    # Example -- handle character reference, no need to override

handle_comment(self, data)
    # Example -- handle comment, could be overridden

handle_data(self, data)
    # Example -- handle data, should be overridden

handle_doctype(self, tag, pubid, syslit, data)
    # Overridable -- handle DOCTYPE

handle_endtag(self, tag, method)
    # Overridable -- handle end tag

handle_proc(self, name, data)
    # Example -- handle processing instructions, could be overrid

handle_starttag(self, tag, method, attrs)
    # Overridable -- handle start tag

handle_xml(self, encoding, standalone)
    # Overridable -- handle XML processing instruction

parse_attributes(self, tag, i, j)
    # Internal -- parse attributes between i and j

parse_cdata(self, i)
    # Internal -- handle CDATA tag, return length or -1 if not te

```

```

parse_comment(self, i)
    # Internal -- parse comment, return length or -1 if not termi

parse_doctype(self, res)
    # Internal -- handle DOCTYPE tag, return length or -1 if not

parse_endtag(self, i)
    # Internal -- parse endtag

parse_proc(self, i)
    # Internal -- handle a processing instruction tag

parse_starttag(self, i)
    # Internal -- handle starttag, return length or -1 if not ter

reset(self)
    # Interface -- reset this instance.  Loses all unprocessed da

setliteral(self, *args)
    # For derived classes only -- enter literal mode (CDATA)

setnomoretags(self)
    # For derived classes only -- enter literal mode (CDATA) till

syntax_error(self, message)
    # Example -- handle relatively harmless syntax errors, could

translate_references(self, data, all=1)
    # Interface -- translate references

unknown_charref(self, ref)

unknown_endtag(self, tag)

unknown_entityref(self, name)

unknown_starttag(self, tag, attrs)
    # To be overridden -- handlers for unknown objects

```

Data and other attributes defined here:

attributes = { }

elements = { }

entitydefs = {'amp': '&', 'apos': ''', 'gt': '>', 'lt': '<', 'quot': '"'}

Functions

test(args=None)

Data

```
amp = <_sre.SRE_Pattern object>
attrfind = <_sre.SRE_Pattern object>
attrtrans = '\x00\x01\x02\x03\x04\x05\x06\x07\x08 \x0b\x0c \x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\
!"#$%&\\"*+,-.\\xcf\\xd0\\xd1\\xd2\\xd3\\xd4\\xd5\\xd6\\xd7\\xd8\\xd9\\xda\\xdb\\xdc\\xdd\\xde\\xdf\\xe0\\xe1\\xe2\\
cdataclose = <_sre.SRE_Pattern object>
cdataopen = <_sre.SRE_Pattern object>
charref = <_sre.SRE_Pattern object>
commentclose = <_sre.SRE_Pattern object>
commentopen = <_sre.SRE_Pattern object>
doctype = <_sre.SRE_Pattern object>
doubledash = <_sre.SRE_Pattern object>
endbracket = <_sre.SRE_Pattern object>
endbracketfind = <_sre.SRE_Pattern object>
endtagopen = <_sre.SRE_Pattern object>
entityref = <_sre.SRE_Pattern object>
illegal = <_sre.SRE_Pattern object>
interesting = <_sre.SRE_Pattern object>
ncname = <_sre.SRE_Pattern object>
newline = <_sre.SRE_Pattern object>
procclose = <_sre.SRE_Pattern object>
procopen = <_sre.SRE_Pattern object>
qname = <_sre.SRE_Pattern object>
ref = <_sre.SRE_Pattern object>
space = <_sre.SRE_Pattern object>
starttagend = <_sre.SRE_Pattern object>
starttagmatch = <_sre.SRE_Pattern object>
starttagopen = <_sre.SRE_Pattern object>
tagfind = <_sre.SRE_Pattern object>
version = '0.3'
xmldecl = <_sre.SRE_Pattern object>
xmlns = <_sre.SRE_Pattern object>
```